GoldenGate Client Xtra Reference Manual

For Macromedia Authorware

integration
new media © Integration New Media Inc 2002 | 2007-09-11

Contents

Contents 3

Methods Reference 5

Global Methods 5

GGStatus 5

GGError 5

GGConnection Methods 6

New 6

SuspendDB 6

ResumeDB 7

GetUserList 7

DisconnectUser 7

GetDBOpenList 8

GetDBList 8

GGDBE Methods 9

New 9

GetStruct 9

GGRecordSet Methods 11

New 11

AddRecord 11

DeleteRecord 11

EditRecord 12

GetField 12

GetPosition 12

GetSelection 13

Go 14

GoFirst 14

GoLast 14

GoNext 15

GoPrevious 15

OrderBy 15

Select 16

SelectAll 17

SelectCount 17

SetCriteria 17

SetField 18

UpdateRecord 19
GGSQLQueryRS Methods 19
New 20
GetField 20
GetPosition 20
GetSelection 21
Go 22
GoFirst 22
GoLast 22
GoNext 23
GoPrevious 23
SelectCount 23
Client-side error codes **25**
Supported SQL grammar **27**

Methods Reference

All the method names and parameters listed below are case insensitive. Upper case characters are sometimes used for better legibility.

Global Methods

GGStatus

Syntax `GGStatus()`

Parameters

Return Success: 0

Failure: error code (<0)

Description Check if the last command used succeeded or failed.

Example

```
Err := GGStatus()
If Err < 0 then
    GoTo(@"NotifyUser")
End if
```

See also GGError, page 5

GGError

Syntax `GGerror(`*`[ErrorCode]`*`)`

Parameters *`ErrorCode`* is the error code.

Return Success: a string describing the error

Failure: error code (<0)

Description Retrieve a description of the error corresponding to *ErrorCode* or to the latest error

Examples

```
Err := GGStatus()
If Err < 0 then
    errormessage := GGerror()
End if
```

See also GGStatus, page 5

GGConnection Methods

In each of the methods below, `rCXinstance` designates the GGConnection instance the method must act on.

New

Syntax `newObject("GGConnection", serverIDString, serverPortNumber, userNameString, userPasswordString)`

Parameters *`serverIDstring`* is the IP number or address of GoldenGate Server. Example: "197.171.19.33" or "QuizServer.bisney.com".

`serverPortNumer` is the IP port number on which the GoldenGate expects queries.

`userNameString` and *`userPasswordString`* are account information used to log onto the server. They must match one of the User Name / Password sets defined on the GoldenGate Server. If the Server accepts anonymous connections, *`userNameString`* can be "GUEST" and *`userPasswordString`* can be "ANONYMOUS".

Description Creates a *`GGConnection`* object. This is the very first method to call in any project that uses GoldenGate. `New` returns a reference to the object created. You typically assign that reference to a global variable. `New` initiates a network connection.

Always check that `New` succeeded by calling `ObjectP`.

Example
```
gCX := NewObject("GGConnection", "194.19.171.40", 2000,
    "PizzaEater", "hunger")
```

See also **Error! Reference source not found.**

New (GGDBE)

SuspendDB

Syntax `SuspendDB(rCXinstance, DBName])`

Parameters *`rCXinstance`* is an instance of the Xtra "GGConnection"

`DBName` is the name of the Database to lock.

Return Success: 0

Failure: error code (<0)

Description Lock a database: regular users can't access to this database anymore except if an administrator unlocks it.

This command is accessible only if the user has administrator privileges.

Examples
```
CallObject (CX,"SuspendDB","Pizza")
```

See also ResumeDB

ResumeDB

Syntax `ResumeDB(rCXinstance, DBName)`

Parameters *rCXinstance* is an instance of script " GGConnection "

DBName is the name of the Database to unlock.

Return Success: 0

Failure: error code (<0)

Description Unlock a database: regular users can access to this database again if it was previously suspended.

This command is accessible only if the user has administrator privileges.

Example
```
CallObject (CX, “ResumeDB”, "Pizza")
```

See also SuspendDB

GetUserList

Syntax `GetUserList(rCXinstance)`

Parameters *rCXinstance* is an instance of script " GGConnection "

Return Success: the list of all users connected to the GoldenGate server in the following format: [[#username :<the user name>, #cID :<the connection ID>],…]

Failure: error code (<0)

Description Get the list of all the users connected to the GoldenGate server.

This command is accessible only if the user has administrator privileges.

Example
```
TheList := CallObject(gCX,”GetUserList”)
If listcount(theList) <> 0 Then
    --Show the list in a message variable
    Message := thelist
Else
    -- error
End if
```

See also DisconnectUser

GetDBOpenList

GetDBList

DisconnectUser

Syntax `DisconnectUser(rCXinstance, cID)`

Parameters *rCXinstance* is an instance of script " GGConnection "

cID is the connection ID to query (this value is retrieved with GetUserList method)

Return Success: 0

Failure: error code (<0)

Description Disconnect a remote user identified by *cID*.

This command is accessible only if the user has administrator privileges.

Example
```
CallObject(gCX, “diconnectUser”, 18)
```

See also GetUserList

GetDBOpenList

Syntax `GetDBOpenList(rCXinstance, cID)`

Parameters *rCXinstance* is an instance of script " GGConnection "

cID is the connection ID to query (this value is retrieved with GetUserList method)

Return Success: List of opened databases in the following format: [<DBName1>,<DBName2>,…]

Failure: error code (<0)

Description Get the list opened by a user for a specific connection.

This command is accessible only if the user has administrator privileges.

Example
```
TheList := CallObject(gCX, “GetDBOpenList”, 18)
If listcount(theList)<> 0 then
    -- display the list in a message variable
    message := thelist
Else
    -- error
End if
```

See also GetUserList

GetDBList

GetDBList

Syntax `GetDBList(rCXinstance)`

Parameters *rCXinstance* is an instance of script " GGConnection "

Return Success: List of databases in the following format: [<DBName1>,<DBName2>,…]

Failure: error code (<0)

Description Get the list of all the available databases on the server.

This command is accessible only if the user has administrator privileges.

Example
```
TheList = CallObject(gCX, “GetDBOpenList”)
If listcount(theList) <> 0  then
    -- display the list in a message variable
    message := theList
```

```
Else
    -- error
End if
```

See also GetUserList

GetDBOpenList

GGDBE Methods

In each of the methods below, `rDBinstance` designates the GGBDE instance the method must act on.

New

Syntax `NewObject("GGDBE", rCXinstance, dBName)`

Parameters *`rCXinstance`* GGConnection instance.

`dbName` name of the database to open.

Description Creates a *`GGDBE`* object. `New` returns a reference to the object created. You typically assign that reference to a global variable.

Always check that `New` succeeded by calling `ObjectP`.

Example
```
gDB := NewObject("GGBDE", "Pizza")
```

See also New (GGRecordset)

New (GGSQLQueryRS)

GetStruct

Syntax `GetStruct(rDBinstance)`

Parameters *`rDBinstance`* is an instance of script "GGDBE"

Return Success: the structure of the opened database in the following format: [[#TableName : <name of the table>, #Fields :[#FieldName : <name of the field >, #FieldType :<type of the field >, #Precision :<data size in bytes>]],…]

Failure: error code (<0)

Description Get the structure of the opened database.

Possible field types are: #CHAR, #VARCHAR, #WCHAR, #SMALLINT, #INTEGER, #FLOAT, #DOUBLE, #BYTE, #LONGINT, #BINARY, #VARBINARY, #DATE, #TIME, #TIMESTAMP.

Examples
```
TheList = CallObject(gDB, “GetStruct”)
If listcount(theList) <> 0 then
    -- display the list in a message variable
    message := theList
Else
    -- error
End if
```

See also

GGRecordSet Methods

In each of the methods below, `rRSinstance` designates the GGRecordSet instance the method must act on.

New

Syntax `NewObject("GGRecordSet", rDBinstance, TableName)`

Parameters *`rDBinstance`* is an instance of the Xtra "GGDBE"

`TableName`: name of the table in remote GoldenGate database to open.

Description Creates a "GGRecordSet" Xtra instance. This is typically done right after `new(GGDBE)` successfully completes. `New` returns a reference on the "GGRecorset" instance that you normally assign to a global variable.

Always check if `New` succeeded by calling `ObjectP`.

Example
```
gRS := NewObject("GGRecordSet", "ingredients")
```

See also New(GGDBE)

AddRecord

Syntax `AddRecord(rRSinstance)`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordSet"

Return Success: 0

Failure: error code (<0)

Description Adds a record to the table. It must be followed by one or more calls to `SetField`s and finally by `UpdateRecord`.

Examples
```
Callobject (gRS, “AddRecord”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
CallObject (gRS, “SetField”,"FirstName", "Eric")
CallObject (gRS, “SetField”"LastName", "Cartman")
CallObject (gRS, “UpdateRecord”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
```

See also EditRecord

SetField

UpdateRecord

DeleteRecord

Syntax `DeleteRecord(rRSinstance)`

Parameters *`RTinstance`* is an instance of the Xtra "GGRecordSet"

return Success: 0

Failure: error code (<0)

Description Deletes the current record. The record is immediately removed from the Server.

Examples
```
CallObject(gRS, “DeleteRecord”)
```

EditRecord

Syntax `EditRecord(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordSet"

Return Success: 0

Failure: error code (<0)

Description Enables the current record to be modified. Calling `EditRecord` is mandatory before calls to `SetField`. To actually send the modifications to the Server, call `UpdateRecord`.

Example
```
CallObject (gRS, “EditRecord”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
CallObject (gRS, “SetField”,"FirstName", "Wendy")
CallObject (gRS, “SetField”, "LastName", "Testaburger")
CallObject (gRS, “UpdateRecord”)
--Error Checking(GGError , GGStatus)
```

See also AddRecord

SetField

UpdateRecord

GetField

Syntax `GetField(`*`rRSinstance`*`, `*`fieldName`* `[,`*`format`*`])`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordSet"

`fieldname` is the name of the field who's value must be retrieved

`format` is a valid formatting pattern for fields of type Integer, Float and Date. All the patterns defined in the *V12-DBE User Manual, Reading Fields of Type String, Integer, Float and Date* are accepted.

Return Success: the value of field `fieldName` from the current record.

Failure: error code (<0)

Description Retrieves the value of field `fieldName` from the current record. No network activity is initiated by `GetField`.

Example
```
val := CallObject(gRS, “GetField”,"price")
val := CallObject(gRS, “GetField”, "price","###9.99$")
val := CallObject(gRS, “GetField”,"date", "dd/mm/yy")
```

See also GetSelection

GetPosition

Syntax `GetPosition(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra " GGRecordSet "

Return Success: position of the current record.

Failure: error code (<0).

Description Retrieves the position of the current record in the selection.

Example

```
p := CallObject(gRS, “GetPosition”)
n := CallObject(gRS, “SelectCount”)
if (p < n) then
   CallObject(gRS, “GoNext”)
else
   --Error handling
end if
```

See also SelectCount

GetSelection

Syntax `GetSelection(`*`rRSinstance`*`, `*`outputType`* `[, `*`from`* `[, `*`numRec`*`]] [, `*`fieldNames`*`] [,`*`fieldDelimiter`* `[, `*`recordDelimiter`*`]])`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordset"

`outputType` is #LITERAL, #LIST or #PROPERTYLIST.

`from` is the number of the first record to retrieve. The default value is 1.

`numRec` is the total number of records to retrieve. The default value is *all records*. If you need to explicitly specify *all records* and do not know the exact number of records in the selection, pass a very large number as `numRec`, or check the size of the selection with `SelectCount`.

`fieldNames` is the lingo list of the names of fields to retrieve (empty list [] means all fields).

`fieldDelimiter` is the character used to delimit fields, if `outputType` is #LITERAL. The default value is TAB. If the specified output type is #LIST or #PROPERTYLIST, `fieldDelimiter` is ignored.

`recordDelimiter` is the character used to delimit records, if the `outputType` is #LITERAL. The default value is RETURN. If the specified output type is #LIST or #PROPERTYLIST, `fieldDelimiter` is ignored.

Return Success: the selection..

Failure: error code (<0).

Description Returns one or more records from the local selection as a string, list or property list. `GetSelection` offers a high degree of versatility thanks to the following parameters. All, but the first one and the second one (the GGRecordSet instance and the output type) are optional.

Example

```
str := CallObject(gRS, “GetSelection”)
aList := CallObject(gRS, “GetSelection”,#LIST, 1, 50)
fld := CallObject(gRS, “GetSelection”,#LITERAL, 1, 10,
    ["LastName", "FirstName"], ",", RETURN)
```

See also GetField

Go

Syntax `Go(rRSinstance, pos)`

Parameters *rRSinstance* is an instance of the Xtra "GGRecordSet"

pos is the number of the destination record in the selection

Return Success: 0.

Failure: error code (<0).

Description Sets the current record to record number `pos`. Regardless of what the current record is at a given instant, invoking `Go(rRSinstance, pos)` sets the current record to the `pos` of the selection.

Example

```
err := CallObject(gRS, “Go”, 25)
```

See also GoFirst

GoLast

GoNext

GoPrevious

GetPosition

GoFirst

Syntax `GoFirst(rRSinstance)`

Parameters *rRSinstance* is an instance of the Xtra "GGRecordSet"

Return Success: 0.

Failure: error code (<0).

Description Sets the first record of the selection as the current record.

Example

```
err := CallObject(gRS, “GoFirst”)
```

See also Go

GoLast

GoNext

GoPrevious

GetPosition

GoLast

Syntax `GoLast(rRSinstance)`

Parameters *rRSinstance* is an instance of the Xtra "GGRecordSet"

Return Success: 0.

Failure: error code (<0).

Description Sets the last record of the selection as the current record.

Example

```
err := CallObject(gRS, “GoLast”)
```

See also Go

GoFirst

GoNext

GoPrevious

GetPosition

GoNext

Syntax `GoNext(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordSet"

Return Success: 0.

Failure: error code (<0).

Description Sets the record next to the current record as the new current record. If there is no next record, the current record remains the same (i.e., the last of the selection) and an error is signaled.

Example
```
err := CallObject(gRS, “GoNext”)
```

See also Go

GoFirst

GoLast

GoPrevious

GetPosition

GoPrevious

Syntax `GoPrevious(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordSet"

Return Success: 0.

Failure: error code (<0).

Description Sets the record preceding the current record as the new current record. If there is no previous record, the current record remains the same (i.e., the first of the selection) and an error is signaled.

Example
```
err := CallObject(gRS, “GoPrevious”)
```

See also Go

GoFirst

GoLast

GoNext

GetPosition

OrderBy

Syntax `OrderBy(`*`rRSinstance`*`, `*`fieldName`*` [, `*`sortOrder`*`] [, `*`fieldName`*` [, `*`sortOrder`*`]]*)`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordSet"

fieldName is the name by which the selection must be sorted.

sortOrder is the sort order by which the selection must be sorted.

Return Success: 0.

Failure: error code (<0).

Description Sorts the selection according to values in field `fieldName`. `sortOrder` is either #ASCENDING or #DESCENDING. The default value is #ASCENDING. The first *fieldname* is mandatory. It defines the primary key. Subsequent *fieldnames* define secondary, third, etc. keys.

`OrderBy` is convenient to call right after setting search criteria (with `SetCriteria`) to determine the sorting order of the resulting records.

`OrderBy` must be followed by `Select` or `SelectAll`.

If you don't call `OrderBy` before `Select or SelectAll`, GoldenGate uses the default index to build the selection the fastest possible. Thus, the resulting selection would be sorted according to that index.

Example
```
CallObject(gRS, “SetCriteria”,["Age", ">", 20])
if GGStatus() <> 0 then GoTo(@"NotifyUser")
CallObject(gRS, “OrderBy”,"FirstName", “Age”, #DESCENDING)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
CallObject(gRS, “Select”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
```

See also SetCriteria

Select

SelectAll

Select

Syntax `Select(`*`rRSinstance`*`)`

Parameters *rRSinstance* is an instance of the Xtra "GGRecordSet"

Return Success: number of records in the selection.

Failure: error code (<0).

Description Performs a selection on the remote table according to criteria set by `SetCriteria` and a sorting order set by `OrderBy`.

`Select` is typically preceded by `SetCriteria` and optionally by `OrderBy`. After performing a `Select`, the set criteria remain valid for further `Select`s

Examples
```
CallObject (gRS, “SetCriteria”, ["country", "=", "Zimbabwe"])
if GGStatus() <> 0 then GoTo(@"NotifyUser")
result := CallObject(gRS, “Select”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
```

See also SetCriteria

SelectAll

SelectAll

Syntax `SelectAll(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordSet"

Return Success: number of records in the selection.

Failure: error code (<0).

Description Selects all the records of the remote table. The order of the selection is the one defined by the last `OrderBy` (see `OrderBy`).

Examples
```
CallObject(gRS, “OrderBy”,"city") -- city is indexed for higher
    performance
if GGStatus() <> 0 then GoTo(@"NotifyUser")
result := CallObject(gRS, “SelectAll”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
```

See also SetCriteria

Select

SelectCount

Syntax `SelectCount(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra "GGRecordSet"

Return Success: number of records in the selection.

Failure: error code (<0).

Description Returns the number of record in the selection.

Example
```
n := CallObject(gRS, “SelectCount”)
if (n > 0) then
   Result := callobject(gRS, “Getselection”, #literal)
else
   alert "No records in the selection. Please modify your
search criteria"
end if
```

See also GetPosition

SetCriteria

Syntax `SetCriteria(`*`rRSinstance`*`, `*`CiteriaList`*`)`

Parameters *`rRSinstance`* is an instance of script "V12RemoteTable"

`CiteriaList` is the list of criteria.

Return Success: 0.

Failure: error code (<0).

Description One call to `SetCriteria` is used to build a search query. Every criterion is stored in the list. You can mix the criteria using Boolean operators at will.

Grammar:

CiteriaList::= {*Criteria* | *Bool-Expression*}

Criteria ::= *Open-Bracket Field-Name , Comparison-op , Expression Close-Bracket*

Bool-Expression ::= *Open-Bracket Bool-factor* , {"AND" | "OR"} , *Bool-factor* [, {"AND" | "OR"} , *Bool-factor*]… *Close-Bracket*

Bool-factor ::= ["NOT"] , {*Criteria* | *Bool-Expression*}

Comparison-op ::= {"<" | ">" | "<=" | ">=" | "=" | "<>" | "START" | "CONTAINS"}

Field-Name ::= name of a valid field .

Expression ::= valid Lingo expression.

Open-Bracket ::= [

Close-Bracket ::=]

For example:

```
CallObject(gRS, “SetCriteria”,[["Price", "<", 1000], "AND",
   [["Destination", "=", "Paris"], "OR", ["Destination", "=",
   "London"]]])
```

would find all tickets for Paris or for London that are cheaper than $1,000.

```
CallObject(gRS, “SetCriteria”,[[["Price", "<", 1000], "AND",
   [["Destination", "=", "Paris"]], "OR", [["Price", "<",
   900], "AND", ["Destination", "=", "London"]]])
```

would tickets for Paris that are cheaper than $1,000 or tickets for London that are cheaper than $900.

Example

```
CallObject(gRS, “SetCriteria”,[["country", "=", "Zimbabwe"], "or",
    ["country", "=", "Ethiopia"]])
if GGStatus() <> 0 then GoTo(@"NotifyUser")
CallObject(gRS, “Select”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
```

See also Select

SetField

Syntax `SetField(rRSinstance, fieldName, value)`

Parameters *rRSinstance* is an instance of the Xtra "GGRecordSet".

fieldname is the name of the field in the current record that will store *value.*

value is the new value to store in *fieldName*.

Return Success: 0.

Failure: error code (<0).

Description Replaces the value in field `fieldName` of the current record by `value`.

Calls to `SetField` must be preceded by a call to `EditRecord` or

`AddRecord`. They must also be followed by a call to `UpdateRecord`.

Value is a string, integer, float date or media (e.g. the media of member(x))

Format for field types #date, #time and #timestamp:

#date: [#year:2000, #month:1, #day:11]

#time: [#hour:4, #minute:30, #second:0]

#timestamp: : [#year:2000, #month:1, #day:11, #hour:4, #minute:30, #second:0]

Note: time is using 24h international standard.

Examples

```
CallObject(gRS, “EditRecord”)
CallObject(gRS, “SetField”,"FirstName", "Wendy")
CallObject(gRS, “SetField”,"LastName", "Testaburger")
CallObject(gRS, “SetField”,"picture", member("picture").media)
CallObject(gRS, “UpdateRecord”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
```

See also EditRecord

AddRecord

UpdateRecord

UpdateRecord

Syntax `UpdateRecord(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of script "GGRecordSet"

Return Success: 0.

Failure: error code (<0).

Description Sends the newly added or modified current record to the Server. `UpdateRecord` must be preceded by a first call to `AddRecord` or `EditRecord`, and a sequence of `SetField`.

If this command is used after an AddRecord, the current selection will be replaced by the new record.

Example

```
CallObject(gRS, ”AddRecord”)
CallObject(gRS, “SetField”,"FirstName", "Eric")
CallObject(gRS, “SetField”,"LastName", "Cartman")
CallObject(gRS, “UpdateRecord”)
if GGStatus() <> 0 then GoTo(@"NotifyUser")
```

See also EditRecord

AddRecord

GGSQLQueryRS Methods

In each of the methods below, `rRSinstance` designates the GGRecordSet instance the method must act on.

New

Syntax `NewObject("GGSQLQueryRS", rDBinstance, SQLQuery, SQLParams)`

Parameters *rDBinstance* is an instance of the Xtra "GGDBE"

SQLQuery: SQL query.

SQLParams: list of SQL query parameters.

Description Creates a "`GGSQLQueryRS`" Xtra instance. This is typically done right after `new(GGDBE)` successfully completes. `New` returns a reference on the "`GGSQLQueryRS`" instance that you normally assign to a global variable.

Always check if `New` succeeded by calling `ObjectP`.

SQL commands SELECT, UPDATE, DELETE and INSERT are supported. The level of complexity of a query depends on the type of database. For more information on the supported grammar of SQL, see appendix 11.

The queries are using parameters (see example for more details).

Example
```
gRS := NewObject("GGSQLQueryRS", "SELECT * FROM ingredients WHERE
    name = ? OR name = ?", [“tomatoe”,”cheese”])
```

See also New(GGDBE)

GetField

Syntax `GetField(rRSinstance, fieldName [,format])`

Parameters *rRSinstance* is an instance of the Xtra " GGSQLQueryRS "

fieldname is the name of the field who's value must be retrieved

format is a valid formatting pattern for fields of type Integer, Float and Date. All the patterns defined in the *V12-DBE User Manual, Reading Fields of Type String, Integer, Float and Date* are accepted.

Return Success: the value of field `fieldName` from the current record.

Failure: error code (<0)

Description Retrieves the value of field `fieldName` from the current record. No network activity is initiated by `GetField`.

Example
```
val := CallObject(gRS, “GetField”,"price")
val := CallObject(gRS, “GetField”,"price","###9.99$")
val := CallObject(gRS, “GetField”,"date", "dd/mm/yy")
```

See also GetSelection

GetPosition

Syntax `GetPosition(rRSinstance)`

Parameters *rRSinstance* is an instance of the Xtra " GGSQLQueryRS "

Return Success: position of the current record.

Failure: error code (<0).

Description Retrieves the position of the current record in the selection.

Example

```
p := CallObject(gRS, “GetPosition”)
n := CallObject(gRS, “SelectCount”)
if (p < n) then
   CallObject(gRS, “GoNext”)
else
   message := "You are already on the last record of the
selection"
end if
```

See also SelectCount

GetSelection

Syntax `GetSelection(`*`rRSinstance`*`, `*`outputType`* `[, `*`from`* `[, `*`numRec`*`]] [, `*`fieldNames`*`] [,`*`fieldDelimiter`* `[, `*`recordDelimiter`*`]])`

Parameters

`rRSinstance` is an instance of the Xtra " GGSQLQueryRS "

`outputType` is #LITERAL, #LIST or #PROPERTYLIST.

`from` is the number of the first record to retrieve. The default value is 1.

`numRec` is the total number of records to retrieve. The default value is *all records*. If you need to explicitly specify *all records* and do not know the exact number of records in the selection, pass a very large number as `numRec`, or check the size of the selection with `SelectCount`.

`fieldNames` is the lingo list of the names of fields to retrieve (empty list [] means all fields).

`fieldDelimiter` is the character used to delimit fields, if `outputType` is #LITERAL. The default value is TAB. If the specified output type is #LIST or #PROPERTYLIST, `fieldDelimiter` is ignored.

`recordDelimiter` is the character used to delimit records, if the `outputType` is #LITERAL. The default value is RETURN. If the specified output type is #LIST or #PROPERTYLIST, `fieldDelimiter` is ignored.

Return Success: the selection..

Failure: error code (<0).

Description Returns one or more records from the local selection as a string, list or property list. `GetSelection` offers a high degree of versatility thanks to the following parameters. All, but the first one and the second one (the GGRecordSet instance and the output type) are optional.

Example

```
str := CallObject(gRS, “GetSelection”)
aList := CallObject(gRS, ‘GetSelection”,#LIST, 1, 50)
fld := CallObject(gRS, “GetSelection”,#LITERAL, 1, 10,
   ["LastName", "FirstName"], ",", RETURN)
```

See also GetField

Go

Syntax `Go(rRSinstance, pos)`

Parameters *rRSinstance* is an instance of the Xtra " GGSQLQueryRS "

pos is the number of the destination record in the selection

Return Success: 0.

Failure: error code (<0).

Description Sets the current record to record number `pos`. Regardless of what the current record is at a given instant, invoking `Go(rRSinstance, pos)` sets the current record to the `pos` of the selection.

Example
```
err := CallObject(gRS, “Go”,25)
```

See also GoFirst

GoLast

GoNext

GoPrevious

GetPosition

GoFirst

Syntax `GoFirst(rRSinstance)`

Parameters *rRSinstance* is an instance of the Xtra " GGSQLQueryRS "

Return Success: 0.

Failure: error code (<0).

Description Sets the first record of the selection as the current record.

Example
```
err := CallObject(gRS, “GoFirst”)
```

See also Go

GoLast

GoNext

GoPrevious

GetPosition

GoLast

Syntax `GoLast(rRSinstance)`

Parameters *rRSinstance* is an instance of the Xtra " GGSQLQueryRS "

Return Success: 0.

Failure: error code (<0).

Description Sets the last record of the selection as the current record.

Example
```
err := CallObject(gRS, “GoLast”)
```

See also Go

GoFirst

GoNext

GoPrevious

GetPosition

GoNext

Syntax `GoNext(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra " GGSQLQueryRS "

Return Success: 0.

Failure: error code (<0).

Description Sets the record next to the current record as the new current record. If there is no next record, the current record remains the same (i.e., the last of the selection) and an error is signaled.

Example
```
err := CallObject(gRS, “GoNext”)
```

See also Go

GoFirst

GoLast

GoPrevious

GetPosition

GoPrevious

Syntax `GoPrevious(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra " GGSQLQueryRS "

Return Success: 0.

Failure: error code (<0).

Description Sets the record preceding the current record as the new current record. If there is no previous record, the current record remains the same (i.e., the first of the selection) and an error is signaled.

Example
```
err := CallObject(gRS, “GoPrevious”)
```

See also Go

GoFirst

GoLast

GoNext

GetPosition

SelectCount

Syntax `SelectCount(`*`rRSinstance`*`)`

Parameters *`rRSinstance`* is an instance of the Xtra "GGSQLQueryRS"

Return Success: number of records in the selection.

Failure: error code (<0).

Description Returns the number of record in the selection.

Example

```
n := CallObject(gRS, “SelectCount”)
if (n > 0) then
    Result := CallObject(cx, “GetSelection”, #literal)
else
   message := "No records in the selection. Please modify your
search criteria"
end if
```

See also GetPosition

Client-side error codes

The GoldenGate methods can return four different types of errors:

- error code -1: This is the generic error code returned by the database manager on the server side. To get a more descriptive error, call GetError(). Example of error message returned by the ODBC database manager: “ODBC Error. SQLState :42S02 Native error : 208 Diagnostic Msg : [Microsoft][ODBC SQL Server Driver][SQL Server] Invalid object name "peoples".”
- Error code –2: A severe error returned by the GoldenGate server. Do not rely on any GoldenGate object any more. Delete your instances and create them anew.
- Error codes–5000 through –4000 : These errors are generated by the GoldenGate client Xtra (see below for details). Example: “Error: -4200. No connection is opened. Cannot proceed. ”.
- Other Error codes are generated by the low-level communication module. They must be treated as fatal errors (abort and restart program).

-4000	Memory allocation error
-4010	Missing parameters to this handler.
-4020	Too many parameters to this handler.
-4050	A Director internal error as occurred. Cannot proceed.
-4055	An internal error as occurred. Cannot proceed.
-4060	Expecting parameter of type integer in the Lingo call.
-4070	Expecting parameter of type string in the Lingo call.
-4080	Expecting parameter of type symbol in the Lingo call.
-4100	Wrong connection mode passed in the Lingo call.
-4110	Username should be longer than 3 characters
-4111	Username should be shorter than xx characters
-4112	Username contains illegal characters
-4120	Unauthorized port number in Lingo call.
-4140	Password should be longer than 3 characters
-4141	Password should not be longer than xx characters
-4142	Password contains illegal characters
-4130	Illegal server address in Lingo call.
-4160	Connection is not opened.
-4170	Asynchronous connection requires a Lingo handler as parameter.
-4180	No callback handler expected in synchronous mode
-4190	Error message is empty
-4200	No connection is opened. Cannot proceed
-4210	Cannot use Unicode services
-4220	An unexpected message was sent by the server
-4230	Apple Text service for Unicode not installed.
-4240	An error occurred during text manipulation. Either the data sent by

	the server were not formatted properly or a problem occurred with the Text Encoding Converter.
-4245	The parameter must be a valid GGConnection script object.
-4250	The parameter must be a valid GGDBE script object.
-4255	No such field in this table definition.
-4260	Field in the record definition has an unknown type
-4265	Wrong output type symbol. Must be one of #Literal, #List or #PropertyList.
-4270	Expecting linear list as parameter in Lingo call.
-4275	Expecting property list as parameter in Lingo call.
-4280	Expression inside the SetCriteria is not well formed.
-4285	Using an undefined operator in a SetCritera call.
-4290	Date value should be either a Lingo expression or a property list.
-4300	Error in the date property list.
-4305	Time value should be expressed as a property list.
-4305	Error in the time property list.
-4315	Time stamp value should be expressed as a property list.
-4320	Error in the time stamp property list.
-4325	OrderBy parameters must be either field names or one of the symbols #ascending or #descending.
-4330	You must call EditRecord or AddRecord before SetField.
-4335	You must call EditRecord or AddRecord before UpdateRecord.
-4345	Unsupported type in SQL parameters list.
-4350	Expecting a media as parameter.
-4355	No fields have been set to update.
-4360	No current record. The selection might be empty or you have tried to read a record that is not there.
-4365	Wrong year value. It should be between 1 and 9999.
-4370	Wrong month value. It should be between 1 and 12.
-4375	Wrong day value. It should be between 1 and 31.
-4380	Wrong hour value. It should be between 0 and 24.
-4385	Wrong minute value. It should be between 0 and 59.
-4390	Wrong second value. It should be between 0 and 59.
-4395	Wrong time format. When hour is 24 the minute and second values should be 0."}

Supported SQL grammar

This grammar is based on Microsoft's ODBC grammar.

Statement

statement ::= create-table-statement
| delete-statement-searched
| drop-table-statement
| insert-statement
| select-statement
| update-statement-searched

create-table-statement ::=
CREATE TABLE base-table-name
(column-identifier data-type [,column-identifier data-type]…)

delete-statement-searched ::=
DELETE FROM table-name [WHERE search-condition]

drop-table-statement ::=
DROP TABLE base-table-name

insert-statement ::=
INSERT INTO table-name [(column-identifier [, column-identifier]...)]
VALUES (insert-value[, insert-value]...)

select-statement ::=
SELECT [ALL | DISTINCT] select-list
FROM table-reference-list
[WHERE search-condition]
[order-by-clause]

update-statement-searched
UPDATE table-name
SET column-identifier = {expression | NULL }
[, column-identifier = {expression | NULL}]...
[WHERE search-condition]

Element

base-table-identifier ::= user-defined-name

base-table-name ::= base-table-identifier

boolean-factor ::= [NOT] boolean-primary

boolean-primary ::= comparison-predicate | (search-condition)

boolean-term ::= boolean-factor [AND boolean-term]

character-string-literal ::= ''{character}…''
(character is any character in the character set of the driver/data source. To include a single literal quote character ('') in a character-string-literal, use two literal quote characters [''''].)

column-identifier ::= user-defined-name

column-name ::= [table-name.]column-identifier

comparison-operator ::= < | > | <= | >= | = | <> | LIKE

comparison-predicate ::= expression comparison-operator expression

data-type ::= character-string-type
(character-string-type is any data type for which the ""DATA_TYPE"" column in the result set returned by SQLGetTypeInfo is either SQL_CHAR or SQL_VARCHAR.)

digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

dynamic-parameter ::= ?

expression ::= term | expression {+|–} term

factor ::= [+|–]primary

insert-value ::=
    dynamic-parameter
    | literal
    | NULL
    | USER

letter ::= lower-case-letter | upper-case-letter

literal ::= character-string-literal

lower-case-letter ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

order-by-clause ::= ORDER BY sort-specification [, sort-specification]...

primary ::= column-name
    | dynamic-parameter
    | literal
    | (expression)

search-condition ::= boolean-term [OR search-condition]

select-list ::= * | select-sublist [, select-sublist]...
(select-list cannot contain parameters.)

select-sublist ::= expression

sort-specification ::= {unsigned-integer | column-name} [ASC | DESC]

table-identifier ::= user-defined-name

table-name ::= table-identifier

table-reference ::= table-name

table-reference-list ::= table-reference [,table-reference]…

term ::= factor | term {*|/} factor

unsigned-integer ::= {digit}

upper-case-letter ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

user-defined-name ::= letter[digit | letter | _]...

Date, Time, and Timestamp Escape Sequences

Notation:

{d 'value'}

{t 'value'}

{ts 'value'}

grammar:

date-time-escape ::=
    date-escape
    | time-escape
    | timestamp-escape

date-escape ::=
    esc-initiator d 'date-value' esc-terminator

time-escape ::=
    esc-initiator t 'time-value' esc-terminator

timestamp-escape ::=
    esc-initiator ts 'timestamp-value' esc-terminator

esc-initiator ::= {

esc-terminator ::= }

date-value ::=
    years-value date-separator months-value date-separator days-value

time-value ::=
    hours-value time-separator minutes-value time-separator seconds-value

timestamp-value ::= date-value timestamp-separator time-value

date-separator ::= -

time-separator ::= :

timestamp-separator ::=
    (The blank character)

years-value ::= digit digit digit digit

months-value ::= digit digit

days-value ::= digit digit

hours-value ::= digit digit

minutes-value ::= digit digit

seconds-value ::= digit digit[.digit…]